目次

★10　オペレーティングシステムの概要（前回のつづき）

★10.4　OS の機能の例

ここでは OS の機能の例として，複数のプログラムを並行して実行する方式，入出力の効率を上げる方式の 2 つを紹介する．

★10.4.1　例 1: 複数のプログラムを並行して実行する

OS は，資源の有効利用のため，複数の処理をなるべく並行して実行しようとする．ここでは，そのような仕組みの例として，マルチプログラミングとタイムシェアリングシステムの二つを紹介する．

マルチプログラミングとは，複数のプログラムを同時にメモリ上に置き，CPU に実行させるプログラムを短い時間間隔で切り替える処理形態 (☆ 1) のことである．複数のプログラムを逐次実行する場合より資源を有効利用してスループットを向上させることをねらったものである．

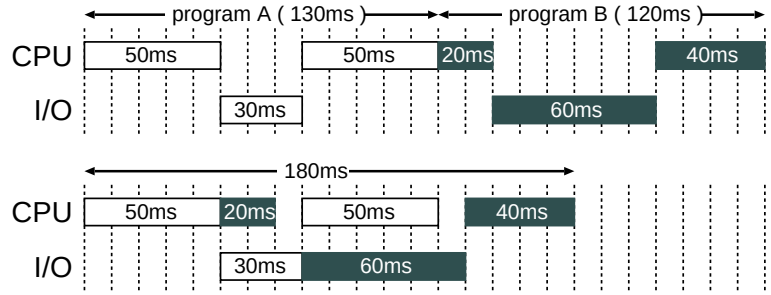
multiprogramming

☆ 1) このような処理形態を**並行処理** (concurrent processing) と言うこともある．並列処理との違いに注意．

例えば，CPU と入出力 (I/O) 装置を使用する 2 つのプログラム A,B の処理時間が下図上段のようになっていたとする．ここでは，CPU，I/O 装置とも一度に 1 つのプログラムだけが使用できるとしている．上段に示すように A の実行後に B を実行する場合，全体では 250ms の時間を要する．そのうち 90ms は CPU が何もしていない**アイドル時間**となっている．マルチプログラミングの形態でこれらを実行する場合，例えば図下段のように実行することで，全体の処理時間を 180ms に，CPU のアイドル時間を 20ms に減らすことができる (☆ 2)．

アイドル時間: idle time.
idle a. 怠惰な，(機械などが) 休止状態である，v. 怠ける，空転する（エンジン等の空転状態を idling という）．idol とは別の語．

☆ 2) ここでは，B より A の方が優先順位が高いとして A の方を先に割り当てているが，逆に A より B の方が優先順位が高ければ B を先に割り当てることになる．このように，資源の割当においては，何らかの基準で優先順位を決める必要のあるケースが多い．



ここで，OS 自身もプログラムであるから，その実行のために CPU の処理時間が必要であることに注意しよう．I/O の際にも OS が働くことになる．上図では簡単のため，それらに要する時間が無視できるほど短いと仮定して省略している．

タイムシェアリングシステムとは，1 台のコンピュータを複数のユーザが共用しているのに，各ユーザがあたかもそのコンピュータを占有しているかのように使えるようにしたものである．各ユーザに一定の短い時間（タイムスライス）を割り当て，割当を次々切り替えていく (☆ 3) ため，ユーザからすると自分専用のコンピュータと対話しているような感覚でコンピュータを使用できる．このような対話型処理の実現により，コンピュータの利用形態は大きく変化した．

Time Sharing System(TSS)

☆ 3) このような処理を**時分割処理**ともいう

★ 10.4.2 例 2: 入出力を効率化する

入出力を効率化する方式を 2 つ紹介する．ただし，ここで紹介するバッファリングやスプーリングは OS だけの機能ではなく，応用ソフトウェア自身が実現していることもある．

一般に，入出力装置を介した処理の速度は遅いため，プログラムの入出力をそのまま処理していると，図左に示すように待ち時間が発生して効率が悪い．これを解決する一つの方法は，入出力データを**バッファ**と呼ばれるメモリ内の領域にいったんためるようにして，入出力の処理と並行してプログラムを実行できるようにする，というものである．これを**バッファリング**という．

スプーリング とは，複数の周辺機器を同時に操作することを意味する．低速な周辺機器とのデータのやりとりを効率よく行うために，データをいったんメモリやディスク装置に書き込み，入出力の処理をプログラム本体の処理と切り離して実行できるようにする方式である．バッファリングではデータを主としてメモリ経由でやりとりするのに対して，スプーリングではディスク上のファイルを経由する場合が多いが，両者は類似した概念のものである．スプーリングの代表例は，プリンタを接続しての印刷処理である．

バッファ: buffer，バッファリング: buffering，スプーリング: spooling (Simultaneous Peripheral Operations On-Line の頭文字をとった SPOOL を動詞化した語)

★ 10.5 カーネルとその機能

オペレーティングシステム (OS) の核となる部分のことを，**カーネル**という．狭義では OS とはカーネルのことを指す．前回説明した OS の役割（資源の管理，コンピュータの利用効率の向上，ハードウェアの抽象化）の大部分を担っている．

カーネルの機能には，主に次のようなものがある．

カーネル: **kernel**．**スーパーバイザ** (supervisor) と呼ぶこともある．前回資料の制御プログラムも同義．

［発展］カーネルの構造／設計思想は，「モノリシックカーネル」と「マイクロカーネル」に大別できる．興味のある人は参考書やウェブ等で調べてみるとよい．Linux はモノリシック，Windows Vista や Mac OS X は（純粋なものではないが）マイクロカーネル．

プロセスの管理とスケジューリング ［詳しくは次回以降の授業で説明予定］

プロセス (☆ 4) とは，実行中のプログラムのことである．この範疇には，次のような仕事が含まれる．

(1) プロセス管理: プロセスを生成，消滅させる．プロセスの状態を把握する．
(2) スケジューリング: マルチプログラミングを採用したシステムでは，全体のスループットが高くなるように，複数のプロセスにプロセッサ（の処理時間）をうまく割り付ける必要がある．このような処理を**スケジューリング** (☆ 5) といい，カーネル内の**スケジューラ**と呼ばれる部分によって行われる．
(3) プロセスの同期と通信の制御: 複数のプロセスが並行して動作する場合，資源を共有するプロセスの間で，それらの処理が矛盾を起こさないようにタイミングを合わせる（同期をとる）必要がある．また，複数のプロセスが協調して動作できるように，**プロセス間通信**を実現しなければならない．

☆ 4) プロセス: process．**タスク** (task) と呼ぶこともある．UNIX 系 OS では，ps コマンドや top コマンドで実行中のプロセスを一覧することができる．

☆ 5) scheduling, scheduler. 実際にプロセッサの割り付けを行う作業のことをディスパッチ (dispatch) といい，それを行うプログラムをディスパッチャ (dispatcher) ということもある．

メモリの管理と仮想記憶の制御 ［詳しくは次回以降の授業で説明予定］

プログラムは，HDD 等の補助記憶装置からメモリ上にロードされてから実行される．メモリ管理とは，プログラムをいつロードするか，メモリ上にどのように割り付けるか，等を決める作業である．

マルチプログラミングシステムでは，複数のプロセスが同時に存在できるように，それらを同時にメモリ上に配置する必要がある．また，仮想記憶 (☆ 6) を採用したシステムでは，その制御も重要な仕事である．

☆ 6) 仮想記憶の方式を用いることで，プロセス全体をロードせず一部しかメモリ上に置いていないときでもその実行が可能となる．

入出力の制御と割り込みの制御 いずれもハードウェア（**デバイス**ともいう）の管理に関するものである．入出力制御とは，様々な入出力デバイスを管理して効率的に使用できるようにすることである．入出力制御のプログラムは，個々のデバイスに特化した部分（**デバイスドライバ** (☆ 7)）と，デバイスの種類によらない共通部分から成る．

割り込みについては次回以降の授業で説明する．

☆ 7) デバイスドライバ: device driver．主要なデバイスのためのドライバはあらかじめカーネルに組み込まれているが，周辺機器メーカが配布するドライバをユーザが自分で組み込むことも多い．

ファイルの管理 ［詳しくは次回以降の授業で説明予定］

OS の機能のうちファイルの管理に関するものは，**ファイルシステム**と呼ばれる．主として補助記憶装置上にファイルやディレクトリを構成し，それらの操作（作成，削除等）やファイルに格納されたデータへのアクセス手段を提供する．

ファイルシステム: file system.

★ 10.6 OS とのやりとり

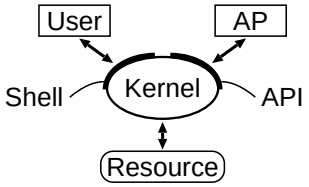
ユーザや応用プログラムが OS とやりとりをするためのインタフェイスについて述べる.

★ 10.6.1 シェル

ユーザは，**シェル** (☆ 8)（または**コマンドプロンプト**）と呼ばれるプログラムを介して OS の機能を利用することができる．龍谷大学の計算機室の Linux 環境のデフォルトでは，ターミナルプログラム（コンソール）上で bash というシェルが動作するようになっている．この場合，ユーザがシェルに対して `./a.out` と入力すると，シェルはこれを「カレントディレクトリの `a.out` という名前のファイルに格納されたプログラムを実行せよ」という指示と解釈して，OS にそのプログラムの実行を要求する.

☆ 8) シェル: shell. 貝殻のこと．あたかもカーネルを取り巻く殻のようであることから名付けられた．UNIX 系 OS では, bash(Bourne Again Shell) 以外にも，csh(C Shell) 等いろいろなものが利用されている．Windows では「コマンドプロンプト」と呼ばれる.

近年では，上述のようなキーボードを用いたインタフェイス (CUI, Character User Interface) のかわりに，GUI(Graphical —) を備えたプログラムを介して OS とやりとりすることも普通になっている (実行ファイルをダブルクリック，等).



★ 10.6.2 API とシステムコール

応用プログラムは，**API** と呼ばれるインタフェイスを介して OS の機能を利用することができる．一般的な OS では，API として OS の様々な機能に対応した関数が定義されており，応用プログラムはそれらを呼び出すことで OS の機能を利用する．UNIX 系 OS では特に，API を通じて呼び出す OS の機能のことを**システムコール**という.

API: Application Programming Interface

例えば，C 言語のプログラムでは，`printf()` や `fprintf()` といった標準ライブラリ関数を呼び出すことで標準出力やファイルに数値や文字を出力することができる．UNIX 系 OS の場合，これらの関数は内部で `write()` というシステムコール関数を呼び出している．この関数がカーネルとやりとりすることで，ファイルにデータを書く処理が行われている (☆ 9).

☆ 9) システムコールとしては，他にもファイル入出力やディレクトリ操作，プロセスの生成や実行終了など様々な処理を行うものが用意されている.